Bob, November 25, 1983

Hope you like this. Cost to do the job right ran higher than expected. The original amount of $27.50 postpaid should cover most of our expenses.

Mike Watters
Zig

Introduction

ZGRASS INFORMATION GROUP

23181 Broadway Avenue
Oakwood Village, OH 44146
(216) 439-3827

To Whom It May Concern:

Copies of this manual and associated ZGrass information were presented to me as a gift from a very good friend. They are being passed on to other friends of mine in an effort to allow those people interested in ZGrass to have a head start on it. The information is being passed on at a cost-of-production basis for the purpose of enlightenment and the expansion of human knowledge. I believe ZGrass is one of the first languages which will allow a nonprofessional computer hacker the ability to interface his ideas, both visual and conceptual, with a computer terminal and do so in a fun, enjoyable environment.

There are several minor differences between this manual and the version which will accompany the early production models of the Viper expansion unit. At a later date, the extended memory and high resolution modification for the Viper (approximately one year) will give the ability to do what this copy of the manual shows that you can do.

The hardware section in the manual deals with the applications for Datamax, yet information pertaining to system controls and checks, disc manipulation, et cetera are directly applicable and will be very useful in your understanding of the versatility of ZGrass. The information about the Paint Program will be useful in understanding how ZGrass was used in designing many of the current arcade games. We have been in contact with and are playing games with the people who own the software rights (Real Time Design) and at this time have no definite commitment from them on production, however, I am confident that when the machine is produced, they will see the light, become greedy, and make software available to the masses, i.e., us.

The manual refers to (a) 256K dynamic screen RAM and (b) high resolution graphics, i.e., 320 x 202 pixels. Both of these differences will be resolved with the high resolution memory expansion board. We are also attempting to make arrangements with several producers of bit pad digitizers, touch pad input devices, mouses [mice(?)], voice synthesizers, voice recognition devices, light pens, et cetera, so that a simple hookup could be made available.

We have had conversations with Jane Veeder, a graphic computer artist from Chicago (see article) who has worked intimately with ZGrass, its designers, and its manufacturers for several years, and have discussed plans of tutorials and/or additional software developments by students of the University Version-1.

There was a delay in the mailing of this information until we could get a commitment from Alternative Engineering that the machine would be produced. Ed Larkin has given me his gentleman's oath as president of Alternative Engineering that the machine will be produced and be first class. We wait as everyone else.

Copyright 1982 Jane Veeder

The background picture is from WARPITOUT—*Veeder's own face jazzed up with the help of ZGRASS, an advanced graphics language. Inset, Veeder and her tools.*

CHICAGO COMPUTER ARTIST ACCELERATES TO WARP SPEED

Just south of the loop, south of the La Salle Street banks, south of the Michigan Avenue hotels, south of the State Street department stores, lives an epic host of poor, underprivileged Chicagoans. Computer artist Jane Veeder is on the leading edge of a new age in electronic expression, yet she lives in the slums on South Halsted Street in a split-level studio apartment.

In that apartment, Veeder and her fellow artist Phil Morton have an impressive array of computer gadgetry, including a Datamax UV-1 Graphics Computer, a Sandin Image Processor, and assorted video equipment. CRT buses shake the building when they roar up Halsted just outside the front door, but they have little effect on the concentration of Veeder and Morton.

Veeder's latest creation, *WARPITOUT,* made an impressive debut at the SIGGRAPH '82 Art Show in Boston this summer. Held annually for the last nine years by the Association for Computing Machinery's Special Interest Group on Computer Graphics, SIGGRAPH is the Academy Awards for computer artists.

In Veeder's own words, *WARPITOUT* is an "interactive computer graphics installation, supporting real-time color graphic processing of a digitized (facial) image of the current player using a menu-driven selection of drawing and processing programs, housed in a video-game cabinet."

Veeder's manifesto further explains the genesis of this computer game cum computer art program: "I had developed a number of generalized real-time computer graphics process program tools that I loved to play with. Some I had adapted into animation sequences, streamlined and stripped of their interactivity; others were still too slow for one-way performance. The recent development of our digitizer offered a wonderful opportunity to present these interactive programs in a menu-driven context for playing with everyone's favorite image . . . themselves."

Burned into eprom and housed in a video-game cabinet, *WARPITOUT* allows the user/player to be Lon Chaney, Salvadore Dali, and Vincent Van Gogh with a digitized image of your own face. It was a big hit at SIGGRAPH, but it raised many questions as to its status as a true work of art.

"Is it art? That's the first question. Let's go on to the next one," Veeder offers. How about an artistic video game? "With *WARPITOUT,* I'm using the universal appeal of your own face as a pretext to indulge in computer graphics more directly than you get to do with a commercial video game, where you're interacting with a finished product in restricted ways. ZGRASS makes possible an artist-integrated project such as *WARPITOUT,* as contrasted to the corporate-designed videogames accomplished by teams working in fragmentary and specialized roles."

All this talk of video games and video game technology is not coincidental. Veeder uses a system that was developed in her neighborhood; video game behemoth Bally is based in Chicago. She has just smartly adapted the technology to other, more personal uses.

A more traditional work of art is *Mon-*

GOTO 16

16

CHICAGO COMPUTER ARTIST...

from page 12

tana, a three-minute color videotape complete with stereo sound. Every year Veeder takes a trip into the western mountains, and *Montana* is an attempt to capture her "love [of] the physical world out there and its attendant information aura."

Montana features a number of forms (mountain, hawk, buffalo, earth, Sears Tower, video camera, and more) made all very simple (in the way the Japanese mean it). Veeder took these simple forms and developed the visual relationships with a fluid program of her own design. The program enables her to draw with any of a collection of "snaps" (screen sections stored as arrays) and tools to make lines, boxes, and other shapes. She worked on the piece for a long time, producing a dynamic, arresting audio-visual experience.

Veeder and her partner Phil Morton are only two of many computer artists living and working in the Chicago area. Tom DeFanti and Dan Sandin of the University of Illinois at Chicago Circle have been very influential, mainly through their own development of new graphics technology. "I'm standing on a whole bunch of people's shoulders," says Veeder.

Veeder came from the world of video synthesis, which she feels is quite wonderful but akin to sex. "It's not too interesting for those watching." She supports herself by doing outside consulting work in the Chicago area, mainly instructing people how to use the ZGRASS language. She also says she's extremely addicted to real-time computer graphics.

"I am almost completely uninterested in still images other than photos for promotion or documentation. Real-time graphic performance resulting in a dynamic visual process is my priority and the motivation for my continuing growth as a programmer."

Living in a slum or not, Veeder is making history in computer art.

[redacted]

FEB 1983 IBM

Language Control Structures for Easy Electronic Visualization

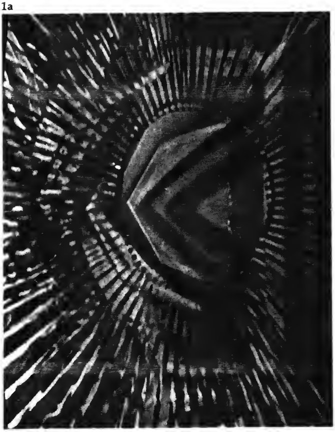
Dr Thomas DeFanti
Electronic Visualization Laboratory
University of Illinois at Chicago Circle
POB 4348
Chicago IL 60680

Control structures are the program- flow manipulation features of the language that you use to beat your computer into submission. BASIC's control structures are embodied in the RUN, GOTO, GOSUB, and RETURN keywords and a few functions, certainly an impoverished set. Highly structured languages like Pascal are rigidly limited to the control structure of subroutines. Lowly structured approaches like assembly language are necessary to implement higher-level languages and real-time systems, because the lack of enforced structure allows an infinite variety of control structures to be used at a cost of great human effort. The execution-speed gain in using assembly language is more due to the efficient building of customized tables and linked lists than to efficiency in adding, subtracting, multiplying, and dividing numbers.

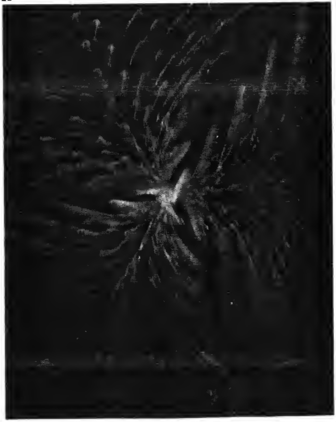
Assembler coding is by no means easy. Note the word "easy": it's important because in one sense it means *"accessible."* In this case, it's your access to complex electronic visualizations.

Electronic visualizations are important because producing and manipulating images, especially animated ones, is a truly multidimensional task which reflects our real-world interactions much more than maintaining an accurate laundry list or printing payroll checks. Producing them demands a lot from software,

1a

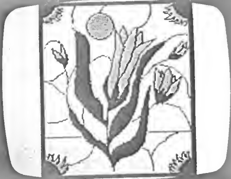


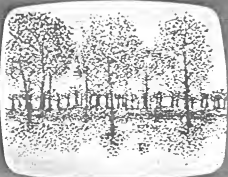
1b



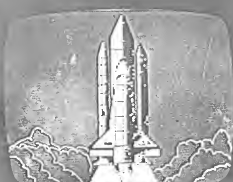
Photos 1a and 1b: *Sample output from the GRASS/Image Processor. Photo 1a was made by Guenther Tetz, and photo 1b by Dan Sandin and the author.*

Circle 55 on inquiry card. ⟶

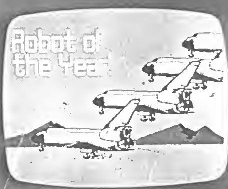




PACIFIC
LIEF ERICSON
ATLANTIC
COLUMBUS



In Space to Stay!

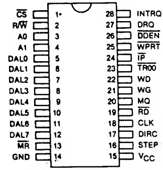

Robot of
the Year

WESTERN DIGITAL
CORPORATION

WD1770/1772 5¼" Floppy Disk Controller/Formatter

FEATURES

- 28 PIN DIP
- SINGLE 5V SUPPLY
- BUILT-IN DATA SEPARATOR
- BUILT-IN WRITE PRECOMPENSATION
- 5¼" SINGLE AND DOUBLE DENSITY
- MOTOR CONTROL
- 128, 256, 512 OR 1024 SECTOR LENGTHS
- TTL COMPATIBLE
- 8 BIT BIDIRECTIONAL DATA BUS
- TWO VERSIONS AVAILABLE
  WD1770 = STANDARD 179X STEP RATES
  WD1772 = FASTER STEP RATES


$\overline{\text{CS}}$ 1
R/$\overline{\text{W}}$ 2
A0 3
A1 4
DAL0 5
DAL1 6
DAL2 7
DAL3 8
DAL4 9
DAL5 10
DAL6 11
DAL7 12
$\overline{\text{MR}}$ 13
GND 14
28 INTRQ
27 DRQ
26 $\overline{\text{DDEN}}$
25 $\overline{\text{WPRT}}$
24 $\overline{\text{IP}}$
23 $\overline{\text{TR00}}$
22 WD
21 WG
20 MO
19 $\overline{\text{RD}}$
18 CLK
17 DIRC
16 STEP
15 V_{CC}

PIN DESIGNATION

DESCRIPTION

The WD1770 is a MOS/LSI device which performs the functions of a 5¼" Floppy Disk Controller/Formatter. It is similar to its predecessor, the WD179X, but also contains a digital data separator and write precompensation circuitry. The drive side of the interface needs no additional logic except for buffers/receivers. Designed for 5¼" single or double density operation, the device contains a programmable Motor On signal.

The WD1770 is implemented in NMOS silicon gate technology and is available in a 28 pin dual-in-line.

The WD1770 is a low cost version of the FD179X Floppy Disk Controller/Formatter. It is compatible with the 179X, but has a built-in digital data separator and write precompensation circuits. A single read line ($\overline{\text{RD}}$, Pin 19) is the only input required to recover serial FM or MFM data from the disk drive. The device has been specifically designed for control of 5¼" floppy disk drives with data rates of 125 KBits/Sec (single density) and 250 KBits/Sec (double density). In addition, write precompensation of 125 Nsec from nominal can be enabled at any point through simple software commands. Another programmable feature, Motor On, has been incorporated to enable the spindle motor automatically prior to operating a selected drive.

Two versions of the WD1770 are available. The standard version is compatible with the 179X stepping rates, while the WD1772 offers stepping rates of 2, 3, 5 and 6 msec.

The processor interface consists of an 8-bit bidirectional bus for transfer of status, data, and commands. All host communication with the drive occurs through these data lines. They are capable of driving one standard TTL load or three "LS" loads.

June, 1983

Copy of page one of the technical data manual for the floppy disk controller (WD1770) which is to be used in the viper extension. This formatter will be introduced to the computer industry at the COMDEX '83 Trade Show at the Los Vegas Convention Center, November 28-December 2.

Operator's
Manual

DATAMAX UV-1
Zgrass GRAPHICS SYSTEM

OPERATOR'S MANUAL

June 1, 1982

Copyright 1982
Real Time Design, Inc.
531 Plymouth Court, Suite 102
Chicago, IL 60605
All Rights Reserved

Glossary

DATAMAX UV-1
Zgrass GRAPHICS SYSTEM

Zgrass GLOSSARY

February 12, 1982

(C) Copyright 1982

Real Time Design, Inc.
531 Plymouth Court, Suite 102
Chicago, IL 60605

All Rights Reserved

Lessons

DATAMAX UV-1
Zgrass LESSONS

October 27, 1981

(C) Copyright 1981
Real Time Design, Inc.

-To delete a whole line, press the TAB key.
-To exit from EDIT, CTRL+E.
-To exit but ignore all changes made in EDIT, press BREAK.
-To insert before the first line, press the HOME key
and type away.
-To insert a NEXTLINE in the middle of a line,
press the HOME key first.

EDIT also allows you to move and copy lines or parts of lines. In order to do this, you must set two pointers, one indicating the start and one past the last character to be copied or moved by positioning the cursor and typing CTRL+S each time. A single quote will appear temporarily to indicate the pointer. (If you make a mistake, type CTRL+T to erase the pointers.) Then position the cursor again to where you want the text to go and type CTRL+D to move the text there or CTRL+F to copy it there. If you confuse things horribly, press BREAK and try EDIT again.

After a while, using EDIT will become easy and you will be forever spoiled.

End of Lesson 3.

MISSING ARGUMENTS:

Often you omit something a command wants as an argument. HELP gives you the arguments in brief for each system command and function, and the Glossary gives information in detail. Sometimes unbalanced parentheses or brackets will confuse the argument scanner, other times it's a misplaced comma or semi-colon. Proofreading is essential in Zgrass since there is not nearly the redundancy present in English or BASIC.

OTHER TYPES OF ERRORS:

There are three other types of errors you will encounter: logic errors, incorrect assumptions, and running out of space and time. We will discuss all of these in detail.

Incorrect assumptions arise from poor documentation or instruction on our part and bad guessing by you. It is impossible to describe everything a programming language does or can do (which is why they're so intriguing, of course) so the best we can do is get you to a level at which you can tell whether it is your problem or a shortcoming of Zgrass when the unexpected happens.

DEBUGGING STEPS:

First, check for syntax errors. If there are none apparent, check that the commands you are using actually do what you expect them to do. If possible, use the command outside a macro in its simplest form. Use numbers instead of expressions since the error will often be in your expression, not in the command, something you should discover as soon as possible.

Always debug in interpretive mode (non-compiled). Anything that works as a macro should work as a compiled macro (if it doesn't, it's our fault unless it's documented).

Macros which make you unhappy do so because they generate errors or don't do what they're supposed to. The former are easier to detect because Zgrass points them out and gives you an error number which you can look up in the Glossary. You can list the program as it is executing with CTRL+X and you will see the last lines executed before the error is generated. You can print the values of variables and

LESSON 9 MORE ON GRAPHICS

Zgrass is a graphics language by design. It provides high-level commands for creating and manipulating visual information. The highest level commands in Zgrass are those operating on arrays called "snaps" (after "snapshots") which are saved parts of the tv screen, in essence. They are defined by the SNAP command and drawn by the DISPLAY command:

```
        STEST=[CLEAR
        BOX 0,0,20,20,3
        BOX 5,0,10,15,2
        BOX -5,0,10,15,1
        SNAP IT,0,0,20,20
        DISPLAY IT,-160%159,-100%100,0;SKIP 0]
```

The last argument of DISPLAY is the DISPLAYMODE. 0 means plop the snap on the screen erasing whatever was there before. DISPLAYMODE 0 works just like COLORMODE 0-3. Now, EDIT STEST so its last line reads:

```
    DISPLAY IT,X=X+$X1,Y=Y+$Y1,0;SKIP 0
```

Plug in a joystick into the leftmost joystick socket and run STEST. You'll notice that you "drag" the edge around. One way to avoid this is to make the snap larger to include some "white space" around it. Change STEST as follows and run it:

```
    SNAP IT,0,0,24,24
```

You can have any pattern stored in a snap as long as it fits in the largest memory segment left. The CORE command will give you a list of the memory segments. The largest snap you can store is about a quarter of the screen. Larger snaps display slower than smaller snaps, of course.

In the drawing program in Lesson 7, we used COLORMODE 5 to flash a crosshair cursor. We can do the same trick with DISPLAYMODE 1:

```
        DRAW=[CLEAR
        LINE 10,0,4
        LINE -10,0,1
        LINE 0,-10,4
        LINE 0,10,1
        SNAP CURS,0,0,20,20
        X=X+$X1;Y=Y+$Y1
        DISP CURS,X,Y,1;DISP CURS,X,Y,1
        IF $T1==1,POINT X,Y,$K1/64+2
        SKIP -3]
```

Two successive DISPLAYMODE 1's will draw and erase the crosshair cursor without affecting the points you have drawn by pulling the trigger.

Zgrass also has a rather complete string manipulation package. Besides the concatenation operator (&), the following are available: ASCII, STRING, BUMP, FORMAT, LEN, LPAD, MATCH, REPLACE, SUBSTR. Some of these are swaps and you should try out these and other swap modules once you find a use for them.

Many of the disk utilities are swap modules. You will find them under the D's in the Glossary.

End of Lesson 10.

Paint Program

DATAMAX UV-1

Zgrass PAINT PROGRAM
USER'S GUIDE

March 12, 1982
Preliminary Release

(C) Copyright 1982
Real Time Design, Inc.
531 Plymouth Court, Suite 102
Chicago, IL 60605
All Rights Reserved

GROUP 1 - BASIC DRAWING BRUSHES

LINES	DRAW/FILL	BOXES
CIRCLES	AIRBRUSH	ELLIPSES

DRAW WITH BOXES

CURSOR BUTTON OPTIONS:

DRAW - SOLID BOX
<YELLOW>

*CHANGE
SIZE MODULE* <WHITE>   <GREEN> BACK TO MENU

<BLUE>
DRAW - OPEN BOX

CURRENT SETTINGS:

COLOR: 1 + MODE: 0 = COLORMODE: 1 WIDTH = 00 HEIGHT = 00

PRESS TRIGGER TO SEE BOX CENTER POSITION

TO CHANGE COLOR AND/OR MODE SETTINGS:

SLIDE CURSOR OFF BOTTOM EDGE OF TABLET AND RETURN IMMEDIATELY; A COLORMAP AND FLASHING "COLOR" CURSOR WILL APPEAR ON TV SCREEN.

o **BOXES**

This is the simplest and most basic of the "rubber stamp" brushes. Selecting this module allows you to define an open or solid box-type brush and then draw with it in a free-hand fashion similar to DRAW. At system start-up and until you change the size of the box brush, a default box will appear on the screen. Note that the center of this flashing box cursor corresponds to the crosshair of the tablet cursor.

OPTION/ BUTTON	ACTIVITY DESCRIPTION
DRAW/ YELLOW	If this random cursor is suitable, you may draw with it by pressing the yellow button on the tablet cursor and proceeding as you did in the DRAW module. If, however, you wish to redefine the dimensions or center position of the cursor, this may be done by selecting one or both of the remaining options.

OPTION/ BUTTON — **CHANGE COLOR/ ACTIVITY DESCRIPTION**

FLIP BETWEEN COLORMAP & IMAGE/ WHITE

As you select new colors for a palette, it's important to be able to see what your image looks like as a new color is substituted. To flip between the screens containing your image and the system colormap, use the white button as a toggle to flip back and forth. NOTE: Even when you've toggled over to the screen displaying your image, the COLOR CONTROL module is still in effect...so that, if you continue to press the yellow button as you move the tablet cursor within the space of the system colormap, you'll be able to see what your image looks like as you continue to change colors.

LIST CURRENT COLORMAP/ BLUE

At first glance, this option doesn't seem to be too terribly useful; however, once you've used the system for awhile, you'll discover that your proficiency at using the system is highly related to your understanding of how colors are created by the system. The information available when you choose this option breaks down the color creation process and illustrates the relationship between the variables (hue and greyvalue) involved in creating a specific color and assigning a numerical color value to that color.

For instance, when you first start up the Paint Program, if you choose the CHANGE COLOR module, the SYSTEM COLORMAP will appear on the TV screen along with the system default drawing colormap. If you choose this option by pressing the white button, the terminal screen will clear and the following chart of information about the four colors contained in the default colormap will appear:

```
             CURRENT COLOR-
COLOR #       MAP VALUE     =  (HUE*8) + GREYLEVEL
-------     ------------       -------   ---------

0 ($L0)          7          =  (  0*8) +     7
1 ($L1)         91          =  ( 11*8) +     3
2 ($L2)        164          =  ( 20*8) +     4
3 ($L3)          8          =  (  1*8) +     0
```

To graphically decipher this information, use the SYSTEM COLORMAP diagram illustration provided in

GROUP 3 - MAKING & USING SNAPS

MAKE SNAP	SHRINK	SNAP DRAW	PATTERN FILL
SCALE SNAP DRAW	EXPAND AND DETAIL	ANIMATE	WORDS

```
DRAW WITH A SNAP
------------------------------------------------------------
                      CURSOR BUTTON OPTIONS:
                      --------------------
                              DRAW
                            <YELLOW>
             CHOOSE
           NEW SNAP <WHITE>          <GREEN> BACK TO MENU

                             <BLUE>
                 INPUT NEW ROTATION (0,90,180,270)


                         CURRENT SETTINGS:
                         ---------------
                 SNAP NAME:  XXXXX   SIZE: 00, 00
                  ROTATION: 0    DISPLAYMODE:  00
   PRESS JOYSTICK TRIGGER FOR CURRENT SNAP POSITION COORDINATES


                      TO INPUT NEW DISPLAYMODE:
                      ------------------------
SLIDE CURSOR OFF BOTTOM EDGE OF TABLET AND RETURN; NEW PRINT WILL
   APPEAR ON THE TERMINAL SCREEN, ASKING FOR NEW DISPLAYMODE.
------------------------------------------------------------
```

o SNAP DRAW

Selecting this module allows you to identify any snap, either in your current work space (user memory) or stored on disk, and then draw with that snap.

When you choose SNAP DRAW, the terminal screen will clear and print will appear, asking for the name of the snap you wish to draw with. After you input the name, you'll be asked whether you want to get it from disk (Y) or not (N). When you've responded, the terminal screen will clear once again and the above-reproduced set of options will appear for your choice.

OPTION ACTIVITY DESCRIPTION

See SYSTEM LIMITATIONS/USER MEMORY sections of GENERAL REFERENCE SECTION for instructions on freeing up user memory.

INPUT 4 TO CHOOSE FOREGROUND AND BACKGROUND FONT COLORS (0-3):

Understanding foreground color is relatively simple...it's the color of the character itself. Background color is a little more complex. The background of a character is the block of blank space surrounding the character that contains pixels not needed to create the characer. Any space between words of text is also considered background. In general, you'll define the background color of your text to be the same as the color of the area on which you are writing, which means the background color of your text will not be noticeable. If, however, you choose to write on a color different from the background color of your text, the effects of doing so will become immediately apparent. Your own experience in dealing with this will best teach you how to use it to your advantage.

In any event, when you input 4, you'll be asked to type in a color variable number (0-3) from the palette colormap currently residing in the system for both the character and the background.

INPUT 5 TO CHANGE PIXEL SPACING BETWEEN CHARACTERS:

This feature allows you to define the number of pixels separating characters of a word. You may spread them further apart or squeeze them together so that they overlap one another. The system default spacing is two pixels between characters.

INPUT 6 TO CHANGE SPACING BETWEEN LINES OF TEXT:

This feature allows you to define the number of lines between lines of text. The system default settings are as follows:

-SMALL FONT: -15
-MEDIUM FONT: -23
-LARGE FONT: -40

Using capital letters, these negative number settings represent the number of lines between the top of the first line and the bottom of the second line.

GROUP 4 - LINE & SNAP ANIMATION

SAVE LINE DRAW	SAVED LINE RE-DRAW	PATH SAVE	PATH MOVE	SAVE PATH SEQUENCE

```
PREREQUISITE:  TO USE THIS MODULE, YOU MUST FIRST HAVE THE
FOLLOWING ITEMS IN USER MEMORY:  A PATH CREATED IN PATH SAVE, THE
SNAPS YOU WISH TO MOVE ALONG THE PATH; IF YOU'RE RUNNING A PATH
SEQUENCE PREVIOUSLY CREATED, YOU MUST ALSO HAVE THE PATH SEQUENCE
AND SNAP GROUP.  USE THE DISK UTILITIES MODULE TO RETRIEVE FILES
STORED ON DISK.  PRESS RETURN TO GO BACK TO THE MENU.

MOVE A SNAP OR SERIES OF SNAPS ALONG A PATH
----------------------------------------------------------------

NAME OF PATH MOVE ANIMATION SEQUENCE?
NAME OF PATH?
NAME OF SNAP GROUP?
IS THIS A NEW GROUP OF SNAPS? (Y OR N)
NAME OF PRIMARY SNAP?
HORIZONTAL SCALE? (default 1)
VERTICAL SCALE? (default 1)
HORIZONTAL OFFSET? (default 0)
VERTICAL OFFSET? (default 0)
DISPLAYMODE? (default 0)
ERASE THE SCREEN? (Y OR N)
DISPLAY SPEED? 1 = 1 STEP AT A TIME
               2 = NORMAL SPEED
               3 = HIGH SPEED

1 STEP AT A TIME:
----------------------------------------------------------------
SNAP NAME:  XXXXXXXXXX   PATH POSITION: 00    SCREEN POSITION: 00
PRESS:  RETURN=NEXT POSITION  C=CHANGE SNAP DISPLAY  G=EXIT STEP
----------------------------------------------------------------
```

o PATH MOVE

Used in conjunction with PATH SAVE, this module allows you to move a snap or series of snaps along the path defined in PATH SAVE. You have the ability to assign a different snap to be displayed at each point of the path or to use a single snap for the entire animation sequence.

Your control over the execution of this type of animation sequence is quite complex and requires that you provide lots of information at the keyboard before you get results on the TV screen. However, by organizing the information in this way, you can save the final result for storage and reproduce the same sequence later. Pay close attention to the PREREQUISITE message printed on the terminal screen when you choose PATH MOVE. You must have all the necessary parts available in user memory. If

```
4M
HELP DOCUMENTATION IS ALSO AVAILABLE ON:
WRITING MACROS (HELP MACRO)       CREATING PICTURES(HELP PIX)
DEVICE USAGE (HELP DEV)           ARITHMETIC PROGRAMMING (HELP FORTRAN)
>>>>>>>>>>>>>TYPE "HELP HELP" TO GET MORE INFORMATION<<<<<<<<<<<<
//  *************************************************************DEV
DEV

DEVICE CONVENTIONS IN GRASS2

     "DEV" IN GRASS GENERALLY MEANS ANYTHING KNOWN TO THE  SYSTEM
THAT  CAN  HOLD  A  VALUE  FROM -32768 TO +32767 (THE LIMITS OF A
16-BIT NUMBER).  BY CONTRAST, FLOATING  POINT  NUMBERS  IN  GRASS
HAVE  MUCH HIGHER RANGES AND TAKE UP MORE BITS AND ARE RELATIVELY
TIME-CONSUMING TO CONVERT TO INTEGER.   16-BIT NUMBERS  ARE  WHAT
THE  VECTOR GENERAL(VG) FEEDS ON, AND FEEDING THE VG IS WHAT GRASS IS
ALL ABOUT AT MACHINE LEVEL.
     SO, THERE'S A BASIC DICHOTOMY IN  GRASS  ARITHMETIC  BETWEEN
FIXED AND FLOATING POINT NUMBERS.  ALL DEV'S ARE FIXED POINT, AND
CAN BE TIED DIRECTLY TO THE 16-BIT REGISTERS IN  THE  VG.   DEV'S
CURRENTLY  INCLUDE  DIALS,  FIXED VARIABLES, FIXED ARRAYS AND, IN
SOME CASES, FIXED POINT EXPRESSIONS.  THUS, IN
              A=B+D0/10
EVERYTHING BUT THE 10 AND THE =,+,/ ARE DEV'S BUT IN
              FA=K+SIN(FB)*FC
ONLY THE K IS A DEV (SEE HELP HELP FOR DEFINITIONS HERE).
     NOW, GRASS  USES  ARITHMETIC  VARIABLES  THAT  ALREADY  HAVE
NAMES.  THIS  ELIMINATES  THE  DECLARE  STATEMENTS  THAT  ARE SO
CUMBERSOME IN MANY PROGRAMMING LANGUAGES AND ALSO VASTLY  SIMPLI-
FIES WHAT GRASS HAS TO DO TO FIGURE OUT WHAT YOU'VE TYPED.  SOME-
DAY, WE MAY DO SOMETHING  ABOUT  LETTING  PEOPLE  HAVE  ARBITRARY
VARIABLE NAMES, BUT FOR THE TIME BEING, THE LEGAL DEV'S ARE:
ARITHMETIC:
     A-Z
     VA-VZ
     WA-WZ
LOCAL ARITHMETIC
     LA-LZ
ARRAY
     AA(...)-AM(...)
SPECIAL ARITHMETIC:
     OA-OH       NORMAL EXCEPT ALSO PUT OUT VOLTAGES TO THE IMAGE
                  PROCESSOR.  0=0 VOLTS, 127=+1 VOLT, -128=-1 VOLT
                  (EXPLAINED MORE LATER).
     SA-SH       CONVERSION TO SINE FUNCTION (EXPLAINED LATER)
     CA-CH       CONVERSION TO COSINE FUNCTION (EXPLAINED LATER)
     MA-MZ AND NA-NZ     LINEAR TIME-BASED VARIABLES
                          (SEE BELOW FOR DETAILED EXPLANATION)
     QA-QZ AND RA-RZ     SINUSOIDAL TIME-BASED VARIABLES

INTERACTIVE CONTROL DIALS, ETC:
     D0-D9      TEN CONTROL DIALS
     S0-S9       TEN SLIDE POTENTIOMETERS
     P0-P3       FOUR MORE CONTROL DIALS
     JX,JY,JZ   JOYSTICK #1
     KX,KY,KZ   JOYSTICK #2
     TX,TY,TZ   TABLET  (TZ=1 FOR PRESS, 0 FOR NEAR & -1 FOR FAR)


     DEV'S CAN BE USED IN ARTIHMETIC EXPRESSIONS (EXCEPT THAT YOU
CANNOT WRITE TO DIALS/POTS/ETC.) AND THEY ALSO CAN BE ATTACHED TO
ROTATIONS, TRANSLATIONS, SCALING AND SO ON.  THIS LATTER CONVENI-
ENCE  ACTUALLY  REQUIRES  SOME  COMPLICATED  SYSTEMS PROGRAMMING,
WHICH YOU SHOULDN'T WORRY ABOUT.  ANYWAY, WHEN BEING USED IN  RO-
TATE, MOVE, SCALE, CUTOFF, SETINT, SETCQ, AND SETORG, THE FOLLOW-
ING RULES APPLY TO DEV'S:


RULE 1: IF THE DEV IS SPECIFIED ALONE, THE SYSTEM  LOOKS  UP  ITS
     VALUE AND USES IT EVERY 1/60 SECOND.  THIS IS WHAT GIVES THE
     INSTANT-SEEMING RESPONSE TO CHANGING DIALS, VARIABLES, ETC.

               EXAMPLE:      SCALE PIX,D0

RULE 2: IF THE DEV IS PART OF  AN  EXPRESSION,  IT  IS  EVALUATED
     ONE-TIME ONLY AND USED AS A CONSTANT:
               EXAMPLE:      SCALE PIX,D9/100+A
     PIX IS SCALED ON THE VALUE OF D9/100+A AT  THE  INSTANT  THE
     SCALE  COMMAND  IS EXECUTED. IF A OR D9 CHANGE, THE SCALE OF
```

```
//  **********************************************************PIX
PIX
```

CREATING PICTURE LISTS IN GRASS

THERE ARE TWO BASIC WAYS TO CREATE PICTURES IN GRASS:

1. WITH THE EDITOR CALL EDIT
USING CALL EDIT, YOU CAN INPUT A PICTURE BY TYPING .DEC FOR THE EXTENSION. THE THING.DEC YOU CREATE IS STORED ON THE DISK AND MAY BE DISPLAYED WITH A GETDSK THING. THE POINTS THAT MAKE UP THE ENDPOINTS OF THE VECTORS YOU WISH TO DRAW MUST BE ENTERED AS FOLLOWS:

```
          X,Y,Z
```

WHERE X, Y, AND Z RANGE FROM +2000 TO -2000. ALL THREE MUST BE SPECIFIED, ONE TO A LINE. THUS, A SQUARE IN THE UPPER RIGHT QUADRANT OF THE SCREEN IS REPRESENTED BY:

```
          0,0,0
          1000,0,0
          1000,1000,0
          0,1000,0
          0,0,0
```

NOTE THAT FIVE POINTS ARE NEEDED TO DRAW FOUR LINES. THE SYSTEM ASSUMES LINES ARE DRAWN FROM THE PRESENT POINT TO THE NEXT POINT SPECIFIED. NOTE ALSO THAT THE ABOVE FORMAT ALLOWS SPACING AS YOU WISH SO FORTRAN I4 FORMAT ON CARDS WITH COMMAS BETWEEN IS ACCEPTABLE TO GRASS.

TO GO TO THE NEXT POINT WITHOUT DRAWING A VECTOR (CALLED A MOVE-NO-DRAW), PUT A "J" ON THE LINES BETWEEN THE POINTS NOT TO BE CONNECTED. TO DRAW A "T", THE FOLLOWING IS ACCEPTABLE:

```
          0,0,0
          0,1000,0
          J
          -500,1000,0
          500,1000,0
```

AND SO ON. THE FIRST POINT IS ALWAYS ASSUMED TO BE A MOVE-NO-DRAW.

AFTER TYPING THE POINTS YOU WANT, EXIT FROM THE EDITOR.

2. WITH PUTPOINT
THE PUTPOINT COMMAND IS USED TO CONSTRUCT PICTURES IN MACROS. IT HAS A SIMILAR FORMAT TO THE DISK FORMAT ABOVE, BUT ENCODES THE MOVE-NO-DRAWS DIFFERENTLY. HELP PUTPOINT SHOULD BE CONSULTED FOR EXACT DETAILS. BRIEFLY, HOWEVER, THE PROCEDURE IS AS FOLLOWS:

A. FIRST, YOU MUST OPEN THE PICTURE NAME. OPEN TELLS GRASS WHAT NAME THE ENSUING PUTPOINT'S, DELPOINT'S AND CLOSE WILL REFER TO. YOU MUST USE AN OPEN BEFORE PUTPOINT'ING.

```
                    OPEN BOX
```

B. THEN YOU TYPE IN, OR HAVE IN A MACRO:

```
                    PUTP 0,0,0,0
                    PUTP 1000,0,0,0
                    PUTP 1000,1000,0,0
                    PUTP 0,1000,0
                    PUTP 0,0,0
```

THIS WILL DRAW THE BOX. NOW YOU MUST TELL THE SYSTEM TO CLOSE

```
                ETC.

//    ****************************************************************
/*

 THERE NO DESCRIPTION AVAILABLE FOR THIS COMMAND AT THIS TIME, OR
 COMMAND NAME HAS BEEN MISSPELLED.  ---- SORRY
//    ****************************************************************
//    **************************************************************** CALL
CALL

SYNTAX:             CALL MNAME

        CALL IS LIKE 'DO' EXCEPT THAT IT REFERENCES THE COMMON AREA ONLY.
CALL, AS WELL AS OTHER COMMANDS, MAY BE USED AS AN ARGUMENT TO RESTART.

EXAMPLES:           CALL MERGE,PIX1,PIX2
                    CALL EDIT
                    RESTART CALL EDIT
NOTE THAT CALL AUTOMATICALLY GETS THE COMPILED VERSION OF THE
MACRO FIRST, IF IT EXISTS.  TO FORCE USING THE MACRO VERSION
(NON-COMPILED) TYPE
                    DO MNAME.MAC 31,3


//    **************************************************************** CUTOFF
CUTOFF

SYNTAX:             CUTOFF PIX,DEV1,DEV2,DEV3,DEV4,DEV5,DEV6

        CUTOFF IS A HARDWARE FUNCTION WHICH CUTS OFF THE
PICTURE EITHER OUTSIDE OR INSIDE A RECTANGULAR BOUNDARY SPECIFIED
BY THE SIX DEV'S. THE DEV'S REPRESENT THE X-HIGH, X-LOW, Y-HIGH,
Y-LOW, Z-HIGH AND Z-LOW BOUNDARIES IN THAT ORDER.  THE DEFAULT VALUES
ARE:
        CUTOFF PNAME,32767,-32768,32767,-32768,32767,-32768
CUTOFF EXPECTS SEVEN VALUES.  IF YOU SPECIFY JUST ONE DEV LIKE
        CUTOFF PNAME,D1
YOU WILL GET D1,D2,D3,D4,D5,D6 AND D7.  YOU SHOULD SET
D1,D3,D5 TO FULL POSITIVE AND D2,D4,D6 TO FULL NEGATIVE TO SEE THE
PIX.  THIS WILL SHOW PNAME AS IT APPEARS WITHIN THE BOUNDARIES
SET BY THE DIALS.  OF COURSE, ANY LEGAL DEV'S ARE OK. (SEE HELP DEV).
THE /R SWITCH WILL SHOW THE PIX OUTSIDE THE BOUNDARIES.
NOTE: CUTOFF SEEMS TO SHIFT WHEN SETP IS USED.
ALSO: CUTOFF IS NOT SOFT'ABLE (SEE HELP SOFT). YOU SHOULD USE THE
WINDOW COMMAND FOR REAL CLIPPING (SEE HELP WINDOW).

SWITCHES:           /R        OUTSIDE INSTEAD OF INSIDE

EXAMPLES:           CUT GLOBE,,,,,,0
        THIS WILL LEAVE THE DEFAULTS FOR ALL BUT Z-LOW
        SO IT WILL SHOW HALF THE GLOBE (ALL POSITIVE Z-POINTS)
                CUT BOX1,D0,=-D0,D0,=-D0,D0,=-D0
                CUT/R BOX2,D0,=-D0,D0,=-D0,D0,=-D0
        THIS SEQUENCE WILL DO A CENTER WIPE FROM BOX1 TO BOX2
        (FIGURE THAT ONE OUT ).

//    ****************************************************************CLEAR

CLEAR

SYNTAX:             CLEAR

        CLEAR SIMPLY CLEARS THE VT05 SCREEN.
//    ****************************************************************  CLOS
CLOSEO


SYNTAX:              CLOSE

        CLOSE ENDS THE PIX PREVIOUSLY SPECIFIED BY OPEN.  CLOSE
ALLOWS THE PIX TO BE PUTDSK'S, IT RELEASES ANY SPACE NOT NEEDED,
AND IT ALLOWS ANOTHER PIX TO BE OPEN'D.

EXAMPLE:

                     OPENO SAM
                     PUTPOI 500,500,500,0
                     PUTPOI -500,500,500,0
                     PUTPOI 0,0,0,0
                     PUTPOI 500,500,500,0
                     CLOSEO
//    **************************************************************** CLIP
CLIP

SYNTAX:             CLIP CLIPEE,CLIPER,CLIPPED
                              WHERE CLIPEE,CLIPER, AND CLIPPED ARE PIX
```

```
ARE CALLING A MACRO FROM SOMEONE ELSE'S AREA. DO'S MAY BE USED
AS OPERANDS TO THE RESTART COMMAND AND ISSUED WITHIN OTHER
MACROS IF DESIRED. ARGUMENTS MAY BE PASSED TO THE MACRO BY THE
USE OF ARG1,ARG2,.... (READ BY THE INPUT COMMANDS INSIDE THE MACRO).
IF ANOTHER MACRO IS CALLED THE OLD ARGUMENTS ARE LOST.

EXAMPLES:          DO FLUFF
                   DO FLUFF.MAC 30,4
                   DO FLUFF.MAC 31,3         (SAME AS "CALL FLUFF")
                   DO FLUFF,10,22,SAM.DEC
                                  (10 IS PASSED TO FIRST INPUT
                                   22 IS PASSED TO SECOND INPUT
                                   THE STRING SAM.DEC IS PASSED TO
                                   THE THIRD INPUT COMMAND, ANY
                                   FURTHER INPUTS WILL GO TO THE
                                   KEYBOARD.)
                                  NOTE THAT PROMPTS ARE SUPPRESSED
                                        UNTIL THE ARGS RUN OUT.

//     ************************************************************ DOLOOP
DOLOOP

SYNTAX:            DOLOOP MNAME1,MNAME2,....    ESOTERIC////
        OR         DOLOOP MNAME
        OR         DOLOOP <....
                          ...
                          ...>

        DOLOOP IS AN ALTERNATIVE TO 'DO' AND IT RUNS THE MACRO
(OR .CPL) AS A BACKROUND JOB. WHEN USING DOLOOP, ALL MACROS
SHOULD BE DOLOOPED SINCE *-LEVEL & NORMAL MACRO LEVEL HAVE
PRIORITY. THINGS REQUIRING INPUT FROM THE TERMINAL IN A DOLOOP'ED
MACRO WILL STOP ALL MACROS, AS WILL SWAP MODULES WHILE THEY ARE
SWAPPING IN. DOLOOPED MACROS MAY HAVE SKIPS, BUT A SKIP AT THE END
BACK TO THE BEGINNING IS ASSUMED. TICK WILL WORK INDIVIDUALLY FOR
EACH MACRO.  LOCAL VARIABLES SHOULD BE USED WHENEVER POSSIBLE.
AT LEAST BE VERY CAREFUL WHEN USING VARIABLES WITH DOLOOPING SO
YOU DON'T HAVE TWO MACROS CHANGING THE SAME VARIABLE UNINTENTIONALLY.
DOLOOPED MACROS ARE EASILY STOPPED BY |C.

SWITCHES:          DOLOOP/E MNAME,EXPR
                    DOES THE MACRO EXPR NUMBER OF TIMES
                   DOLOOP/V MNAME,VAR
                    DOES THE MACRO UNTIL #OF TIMES = VAR

EXAMPLES:          DOLOOP <D=D+D5/32>
                   DOLOOP SAM,FIRED,COMPACT

NAMED DOLOOPED MACROS MAY BE INDIVIDUALLY CANCELLED BY UNLOOP.

//     ************************************************************
/*
 THERE IS NO DESCRIPTION FOR THIS COMMAND AT THIS TIME, OR
COMMAND HAS BEEN MISSPELLED.  ----SORRY.
//     ************************************************************
//     ************************************************************ EXECUT
EXECUTE

SYNTAX:            EXECUTE BNAME                   ESOTERIC/////
        OR         EXECUTE BNAME,ARG1,ARG2,.......

        EXECUTE IS TO .CPL (COMPILED) MACROS WHAT "DO" IS TO
NORMAL MACROS.  THE COMPILED MACRO IS GOTTEN FROM THE DISK (YOUR AREA
UNLESS SPECIFIED IN. XX,YY  LOGIN CONVENTIONS) UNLESS IT'S ALREADY
IN CORE, AND THEN THE COMPILED MACRO IS RUN.  THERE IS NO
EQUIVALENT TO GOTO FOR COMPILED MACROS, BY THE WAY.
        |C AND |S CAN STOP EXECUTION OF COMPILED MACROS.
//     ************************************************************EDIT
EDIT

SYNTAX:            EDIT MNAME

        EDIT IS AN IN-CORE QUICKY EDITOR WHICH WORKS ONLY ON MACROS.
IT IS FAST, SMALL, AND HAS THE ADVANTAGE THAT YOU DO NOT HAVE TO RESTART
GRASS EACH TIME YOU WANT TO USE IT.  MOREOVER, IT CAN BE ENTERED AND
RE-ENTERED VERY QUICKLY.
        IT'S MAJOR DISADVANTAGE IS THAT IT DOES NOT DO EVERYTHING
THAT CALL EDIT DOES.  FOR INSTANCE, YOU TYPE |C TO GET OUT OF THIS
EDIT, AND THE MACRO IS UPDATED ONLY IN CORE, AND NOT ON THE DISK.
FOR THIS REASON, WE HAVE IMPLEMENTED A NEW PUTDSK SWITCH--PUTDSK/D--
WHICH PUTS THE FILE ON THE DISK, FIRST CREATING A BACKUP
(.BAK) IF THE FILE ALREADY EXISTS ON DISK.  THEREFORE, AS SOON AS
YOU HAVE A WORKING VERSION OF A MACRO, PUTDSK/D IT OR YOU WILL LOSE
IT UPON RESTART.  ONE MORE THING:  THIS EDIT WILL NOT CREATE MACROS,
IT WILL ONLY WORK ON MACROS THAT EXIST ON DISK OR IN CORE ALREADY.
        EDIT WILL FETCH A MACRO FROM THE DISK IF IT IS NOT ALREADY
IN CORE, BY THE WAY.
>>>>>>>>EDIT COMMANDS:
```

```
                OR THE COMMAND IS NOT SMART ENOUGH TO ACCEPT
                FULL-BLOWN DEV'S FOR SOME REASON.

>>>>>>>EXPR     IS A MIX OF NUMBERS,DEVS AND ARITHMETIC OPERATORS WHICH
                ALWAYS EVALUATES TO A SINGLE NUMBEP.
                THE OPFRATORS ARE +,-,*,/.
                EXAMPLFS:       A
                                D7
                                200
                                A+(D7/200)*35-B-(K/17)
                THERE IS OPERATOR PRECEDENCE WHEN EXPRESSIONS
                ARE EVALUATED I.E., EXPONENTIATION IS DONE FIRST
                (ONLY FOR USE WITH FLOATING POINT NUMBERS),
                MULTIPLICATION AND DIVISION ARE DONE NEXT, FINALLY
                ADDITION AND SUBTRACTION ARE PREFORMED.
                PARENTHESIS CAN BE INSERTED APOUND PARTS OF AN
                EXPRESSION SO THAT DESIRED OPERATIONS WILL BE
                PERFORMED FIRST.  THUS THE FOLLOWING EXPRESSION
                (52+8)/2+12 EQUALS 42.
                THERE IS A MOD FUNCTION "%" SO 10%4 EVALUATES TO 2.
        SEE HELP FORTRAN FOR MORE DETAILS ON ARITHMETIC STATEMENTS.

>>>>>>>$VAR     IS A STRING VARIABLE AND HOLDS TEXT.
                STRING MANIPULATION IS COMPLICATED EXCEPT
                FOR THIS CASE:
                        PROM "WHAT'S THE PIX NAME"
                        INPUT $A
                        GETDSK $A
                        ROTATE $A,X,D0,D1
                THIS EXAMPLE USES $A AS A DUMMY SO MACROS CAN BE GENERAL
                PURPOSE.  SEE HELP MACROS FOR MORE DETAILS.

DOCUMENTATION CONCERNING SYNTAX FOR A PARTICULAR COMMAND MAY
BE RETRIEVED BY TYPING HELP FOLLOWED BY THE COMMAND NAME:
    HELP ROTATE
WILL GET INFORMATION ON THE ROTATE COMMAND.

THERE ARE ALSO SYSTEM MACROS FOR MOST COMMANDS SO IF YOU DO
NOT UNDERSTAND THE DOCUMENTATION IN HELP, A MACRO WILL STEP YOU
THROUGH MOST COMMANDS.   CALL CLIP, FOR INSTANCE, WILL
RUN YOU THROUGH THE CLIP COMMAND.

IF YOU GET AN ERROR, TYPE A "?" FOLLOWED BY A CARRIAGE RETURN
(AGAIN, REFERRED TO AS <CR>).  THE ERROR MESSAGE WILL BE PRINTED OUT.

COMMANDS MARKED "(STRING MANIPULATION)" AND "     ESOTERIC/////"
ARE BEST SKIPPED BY NOVICES.

IF THE HELP DOCUMENTATION DOES NOT MAKE SENSE, PLEASE MAKE A
LOT OF NOISE ABOUT IT.  IT'S HARD TO GUESS WHAT YOU DON'T UNDERSTAND.

THE HELP DOCUMENTATION IS MOSTLY FOR SYNTAX QUESTION.  SOME ADDITIONAL
CLUES ON HOW TO WRITE GRASS PROGRAMS ARE CONTAINED IN HELP FORTRAN,
HELP DEV, HELP PIX, AND HELP MACROS.

NOTE THAT THE CNTL W BUTTON WILL HALT A HELP PRINTOUT UNTIL YOU HIT
CNTL Q.  YOU TYPE CNTRL Q AND THE OTHER CONTROL FUNCTIONS BY HOLDING
DOWN THE CTRL BUTTON AND TYPING THE LETTER KEY.  IT WORKS JUST LIKE
A SHIFT KEY ON A TYPEWRITER.

THE CNTL S BUTTON WILL HALT EXECUTION OF A MACRO AND PUT THE USER
IN THE #-SIGN MODE. IN THIS MODE THE USER CAN PERFORM ANY OF THE
SYSTEM COMMANDS.  TO RETURN TO MACRO EXECUTION, TYPE 'RESUME'.

REMEMBER TO USE THE CNTL C BUTTON IF THINGS GET OUT OF HAND, AND THAT
THE WORST YOU CAN DO IS STALL THE COMPUTER.

//     ***************************************************************
/*
 THERE IS NO DESCRIPTION AVIALABLE FOR THIS COMMAND AT THIS TIME, OR
COMMAND HAS BEEN MISSPELLED.  ----SORRY.
//     ****************************************************************
//     **************************************************************** IF
IF

SYNTAX:         IF VAR OPR EXPR,COMMAND
        OR      IF FS# OPR EXPR,COMMAND
        OR      IF $VAR OPR ALPHA-EXPR,COMMAND

        IF IS THE SYSTEM'S CONDITIONAL FOR USE WITHIN MACROS. THE
CRYPTIC SYNTAX ABOVE IS BECAUSE THE IF STATEMENT IS SO FLEXIBLE. DEV
AND EXPR ARE DEFINED IN "HELP HELP"
OPR CAN BE    EQ, NE, LE, LT, GE, GT, =, OR # (NOT EQUALS)
FS# IS FS0 - FS15        FOR FUNCTION SWITCHES (0=OFF, 1=ON)
$VAR IS $A - $Z
ALPHA-EXPR IS EITHER $A-$Z OR 'ANY STRING IN QUOTES'. IF ONE OF THE
        CHARACTERS IS A "*", THEN ALL STRINGS MATCHING THE CHARACTERS UP TO
        THE "*" WILL SATISFY THE EQ CONDITION.
```

INTRODUCTION

THIS PROGRAM IS DESIGNED TO HELP YOU LEARN ABOUT THE GRAPHICS SYMBIOSIS SYSTEM (GRASS). THE INTENTION IS TO PROVIDE YOU WITH THE FUNDAMENTAL KNOWLEDGE AND SKILLS NECESSARY TO ALLOW YOU TO USE THE THE SYSTEM. IT IS NOT INTENDED TO PROVIDE YOU WITH A COMPLETE DESCRIPTION OF ALL THE POSSIBLE WAYS IN WHICH THE SYSTEM COULD BE USED. THAT WOULD BE IMPOSSIBLE BECAUSE THE USES OF THE SYSTEM ARE LIMITED ONLY BY YOUR OWN IMAGINATION.

THERE ARE THREE PARTS TO THIS PROGRAM. DEPENDING ON YOUR PRESENT KNOWLEDGE OF THE SYSTEM, YOU MAY WISH TO GO THROUGH ALL, OR ONLY SOME, OF THESE. THE FIRST PART WILL TEACH THE COMMANDS, AND ALLOW PRACTICE IN THE SKILLS, NECESSARY TO CREATE PICTURES. THE SECOND PART WILL DO THE SAME FOR PICTURE MANIPULATION. THE THIRD PART WILL BE ON "MACRO" WRITING. EACH PART WILL ASSUME KNOWLEDGE OF THE PRECEEDING PARTS.

PART 1 PICTURE CREATION

OBJECTIVES: UPON COMPLETION OF THIS PART THE USER WILL...

1. BE ABLE TO DRAW PICTURES BY COMBINING GRASS COMMANDS, VARIABLES, AND PROGRAM LOOPS.